In [1]:
```
# Data Wrangling
import numpy as np
import pandas as pd

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt

# Date Functionality
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter

# Statistical analysis
from scipy.stats import ttest_ind


# Remove Warnings
import warnings
warnings.filterwarnings("ignore")
```

Examining Transaction Data.

In [2]:
```
Data_transaction = pd.read_excel("E:\Deakin\QVI_transaction_data.xlsx")
Data_transaction
```

Out[2]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g
1	43599	1	1307	348	66	CCs Nacho Cheese 175g
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g
...	...	...	...	...	...	...
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g
264832	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g
264833	43410	272	272379	270187	51	Doritos Mexicana 170g
264834	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g
264835	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g

264836 rows × 8 columns

To Check Missing Data

In [3]:
```
Data_transaction.isnull().sum()
```

Out[3]:
```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
dtype: int64
```

- No Missing Data in the datasets.

In [4]:
```
# Look for duplicated TXN_ID

Data_transaction[Data_transaction.duplicated(['TXN_ID'])].head()
```

Out[4]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PRO
42	43605	55	55073	48887	113	Twisties Chicken270g	
377	43475	7	7364	7739	20	Doritos Cheese Supreme 330g	
419	43391	12	12301	10982	93	Doritos Corn Chip Southern Chicken 150g	
476	43351	16	16427	14546	81	Pringles Original Crisps 134g	
511	43315	19	19272	16683	31	Infzns Crn Crnchers Tangy Gcamole 110g	

In [5]:
```
# Select the first duplicated TXN_ID

Data_transaction.loc[Data_transaction['TXN_ID'] == 48887, :]
```

Out[5]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD
41	43605	55	55073	48887	4	Dorito Corn Chp Supreme 380g	
42	43605	55	55073	48887	113	Twisties Chicken270g	

Categorise Numeric and Categorical Data

In [6]:
```
Data_transaction_numerics_only = Data_transaction.select_dtypes(include=np.nu
Data_transaction_cat = set(Data_transaction.columns) - set(Data_transaction_n
```

In [7]:
```
print("Numeric Columns:\n",list(Data_transaction_numerics_only))
print("Categorical Columns:\n",list(Data_transaction_cat))
```

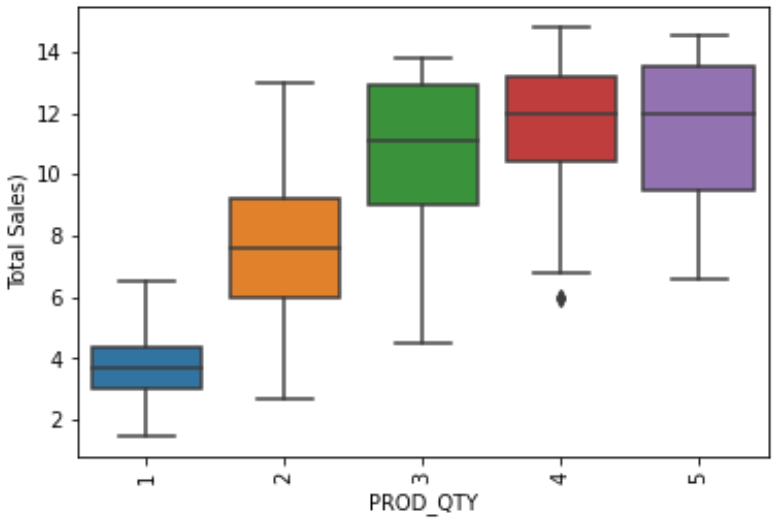
```
Numeric Columns:
 ['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR', 'PROD_QTY', 'TO
T_SALES']
Categorical Columns:
 ['PROD_NAME']
```

To Check Outliers and Treat them

In [8]:
```
def remove_outlier_IQR(df):
    Q1=df.quantile(0.25)
    Q3=df.quantile(0.75)
    IQR=Q3-Q1
    df_final=df[~((df<(Q1-1.5*IQR)) | (df>(Q3+1.5*IQR)))]
    return df_final
```

In [9]:
```
df_outlier_removed=remove_outlier_IQR(Data_transaction.TOT_SALES)
df_outlier_removed=pd.DataFrame(df_outlier_removed)
ind_diff=Data_transaction.index.difference(df_outlier_removed.index)

for i in range(0, len(ind_diff),1):
    df_final=Data_transaction.drop([ind_diff[i]])
    Data_transaction =df_final

sns.boxplot(y='TOT_SALES', x='PROD_QTY',data=Data_transaction)
plt.xticks(rotation=90)
plt.ylabel('Total Sales)')
```

Out[9]:
```
Text(0, 0.5, 'Total Sales)')
```



In [10]:
```
print("Shape of dataset after treating outliers:",Data_transaction.shape)
```

```
Shape of dataset after treating outliers: (264258, 8)
```

- Comparing the two boxplots, it can be seen that the outliers were removed.

In [11]:
```
# Convert Date column into DATE format
origin = pd.Timestamp("30/12/1899")
Data_transaction["DATE"] = Data_transaction["DATE"].apply(lambda x: origin +
Data_transaction
```

Out[11]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g
5	2019-05-19	4	4074	2982	57	Old El Paso Salsa Dip Tomato Mild 300g
...	...	...	...	...	...	...
264831	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g
264832	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g
264833	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g
264834	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g
264835	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g

264258 rows × 8 columns

In [12]:
```
Data_transaction['PROD_NAME'].value_counts()
```

Out[12]:
```
Kettle Tortilla ChpsHny&Jlpno Chili 150g    3285
Kettle Mozzarella   Basil & Pesto 175g      3280
Tyrrells Crisps     Ched & Chives 165g      3264
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g    3260
Cobs Popd Sea Salt  Chips 110g              3259
                                            ...
Woolworths Medium   Salsa 300g              1430
RRD Pc Sea Salt     165g                    1429
French Fries Potato Chips 175g              1418
NCC Sour Cream &    Garden Chives 175g      1416
WW Crinkle Cut      Original 175g           1410
Name: PROD_NAME, Length: 114, dtype: int64
```

In [13]:
```
# The customer only wants insights on chips category, hence, we eliminate all
Data_transaction = Data_transaction[Data_transaction["PROD_NAME"].str.contain
Data_transaction
```

Out[13]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g
6	2019-05-16	4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g
...	...	...	...	...	...	...
264831	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g
264832	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g
264833	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g
264834	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g
264835	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g

246204 rows × 8 columns

In [14]:
```
Data_transaction['PROD_NAME'].value_counts()
```

Out[14]:
```
Kettle Tortilla ChpsHny&Jlpno Chili 150g    3285
Kettle Mozzarella   Basil & Pesto 175g      3280
Tyrrells Crisps     Ched & Chives 165g      3264
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g    3260
Cobs Popd Sea Salt  Chips 110g              3259
                                            ...
Sunbites Whlegrn    Crisps Frch/Onin 90g    1432
RRD Pc Sea Salt     165g                    1429
French Fries Potato Chips 175g              1418
NCC Sour Cream &    Garden Chives 175g      1416
WW Crinkle Cut      Original 175g           1410
Name: PROD_NAME, Length: 105, dtype: int64
```

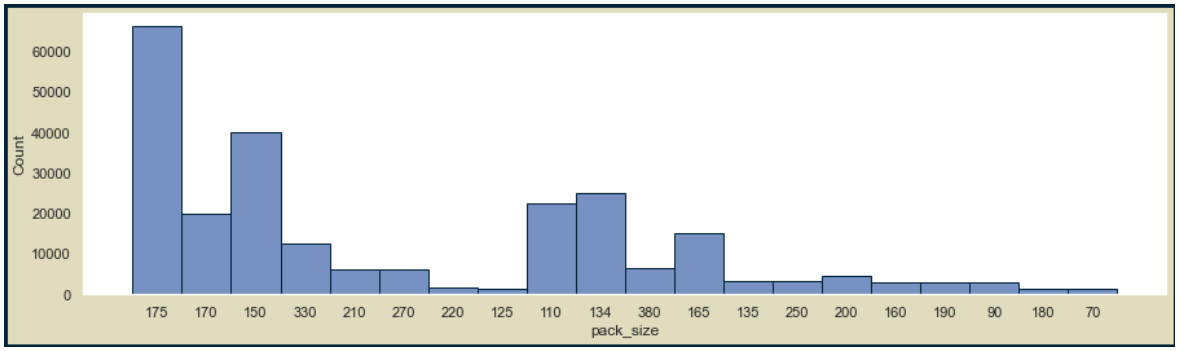
In [15]:
```
# Extracting pack size from the Product
import re
def find_number(text):
    num = re.findall(r'[0-9]+',text)
    return " ".join(num)
Data_transaction['pack_size']=Data_transaction['PROD_NAME'].apply(lambda x: f
Data_transaction
```

Out[15]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g
6	2019-05-16	4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g
...	...	...	...	...	...	...
264831	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g
264832	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g
264833	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g
264834	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g
264835	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g

246204 rows × 9 columns

In [16]:
```
# Histogram showing the number of transactions by pack size

fig = plt.figure(figsize = (15,4), linewidth=5, edgecolor="#04253a", facecolo

sns.set(rc = {'figure.figsize':(15,4)})
ax = sns.histplot(data=Data_transaction, x="pack_size",  edgecolor="#04253a")

ax.set_facecolor("#ffffff")
```



- 175g is the highest selling pack size for the chips followed by 150g.

In [17]:
```
# Create column for brand names
Data_transaction['Brand Name'] = Data_transaction['PROD_NAME'].str.split(' ')
```

In [18]:
```
# Check for any duplication or similar brands
Data_transaction['Brand Name'].value_counts()
```

Out[18]:
```
Kettle        41141
Smiths        27340
Pringles      25052
Doritos       21975
Thins         14049
RRD           11880
Infuzions     11035
WW            10320
Cobs           9669
Tostitos       9443
Twisties       9420
Tyrrells       6428
Grain          6265
Natural        6037
Cheezels       4583
CCs            4551
Red            4427
Dorito         3175
Infzns         3138
Smith          2963
Cheetos        2926
Snbts          1576
Burger         1564
Woolworths     1516
GrnWves        1465
Sunbites       1432
French         1418
NCC            1416
Name: Brand Name, dtype: int64
```

- Some Brands are similar like RRD and Red Rock Deli , WW and Woolworths,NCC and Natural Chip Company etc.Let us combine them together as they are a single unit.

In [19]:
```
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.replace('
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.replace('
```

```
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.replace('
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.replace('
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.replace('
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.replace('
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.replace('
```

In [20]:
```
Data_transaction['Brand Name'].value_counts()
```

Out[20]:
```
Kettle       41141
Smiths       27340
Pringles     25052
Doritos      21975
RRD          16307
Thins        14049
WW           11836
Infuzions    11035
Cobs          9669
Tostitos      9443
Twisties      9420
Tyrrells      6428
Grain         6265
Natural       6037
Cheezels      4583
CCs           4551
Dorito        3175
Infzns        3138
Smith         2963
Cheetos       2926
Snbts         1576
Burger        1564
GrnWves       1465
Sunbites      1432
French        1418
NCC           1416
Name: Brand Name, dtype: int64
```

- Combined the similar brands.

Examining customer data

In [21]:
```
Data_customer = pd.read_csv("E:\Deakin\QVI_purchase_behaviour.csv")
Data_customer
```

Out[21]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream
4	1005	MIDAGE SINGLES/COUPLES	Mainstream
...	...	...	...
72632	2370651	MIDAGE SINGLES/COUPLES	Mainstream
72633	2370701	YOUNG FAMILIES	Mainstream
72634	2370751	YOUNG FAMILIES	Premium
72635	2370961	OLDER FAMILIES	Budget
72636	2373711	YOUNG SINGLES/COUPLES	Mainstream

72637 rows × 3 columns

To check for null Values in the data.

In [22]:
```
Data_customer.isnull().sum()
```

Out[22]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

Categorise Numeric and Categorical Data

In [23]:
```
Data_customer_numerics_only = Data_customer.select_dtypes(include=np.number)
Data_customer_cat = set(Data_customer.columns) - set(Data_customer_numerics_o
```

In [24]:
```
print("Numeric Columns:\n",list(Data_customer_numerics_only))
print("Categorical Columns:\n",Data_customer_cat)
```

```
Numeric Columns:
 ['LYLTY_CARD_NBR']
Categorical Columns:
 {'LIFESTAGE', 'PREMIUM_CUSTOMER'}
```

In [25]:
```
# Merging the two dataframes
df4 = pd.merge(Data_transaction,Data_customer)
df4
```

Out[25]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	P
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips 160g	
3	2019-03-09	1	1307	347	54	CCs Original 175g	
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
...	...	...	...	...	...	...	
246199	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
246200	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	
246201	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	
246202	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
246203	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	

246204 rows × 12 columns

In [26]:
```
# To check if some customers were not matched on by checking for nulls.
df4.isnull().sum()
```

Out[26]:
```
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
pack_size           0
Brand Name          0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [27]:
```
pd.date_range(start = '2018-07-01', end = '2019-06-30').difference(df4['DATE'
```

Out[27]:
```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

We have a missing date on Christmas Day. This makes sense because most retail stores are closed that day.

In [28]:
```
# Create a new dataframe which contains the total sale for each date

df5 = pd.pivot_table(df4, values = 'TOT_SALES', index = 'DATE', aggfunc = 'su
df5.head()
```

Out[28]:

	TOT_SALES
DATE	
2018-07-01	4920.1
2018-07-02	4877.0
2018-07-03	4954.7
2018-07-04	4968.1
2018-07-05	4682.0

In [29]:
```
df6 = pd.DataFrame(index = pd.date_range(start = '2018-07-01', end = '2019-06
df6['TOT_SALES'] = 0
len(df6)
```

Out[29]:
```
365
```

In [30]:
```
z = df5 + df6
z.fillna(0, inplace = True)
```

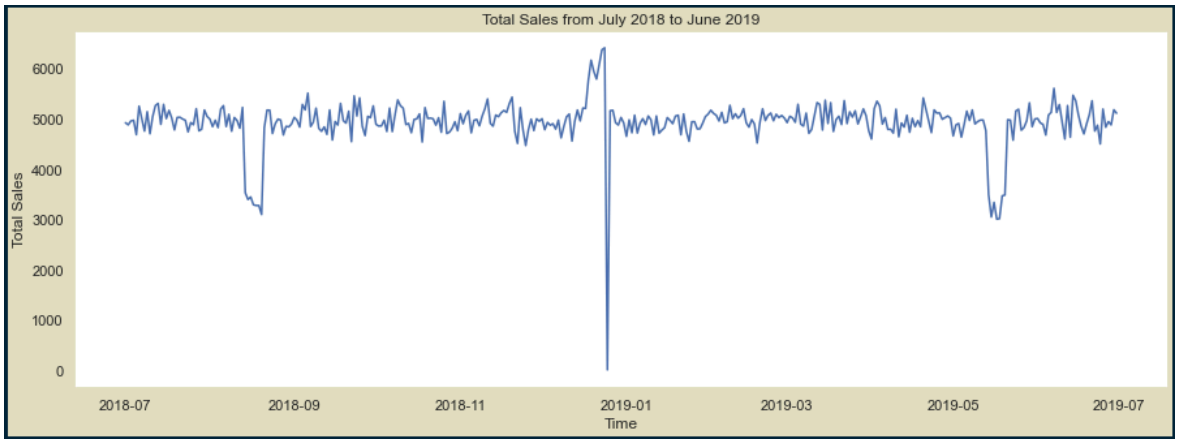
In [31]:
```
z.index.name = 'Date'
z.rename(columns = {'TOT_SALES': 'Total Sales'}, inplace = True)
z.head()
```

Out[31]:

Date	Total Sales
2018-07-01	4920.1
2018-07-02	4877.0
2018-07-03	4954.7
2018-07-04	4968.1
2018-07-05	4682.0

In [32]:
```
timeline = z.index
graph = z['Total Sales']



fig, ax = plt.subplots(figsize = (15, 5), linewidth=5, edgecolor="#04253a", f

ax.plot(timeline, graph)

ax.set_facecolor("#ffffff")

date_form = DateFormatter("%Y-%m")
ax.xaxis.set_major_formatter(date_form)
plt.title('Total Sales from July 2018 to June 2019')
plt.xlabel('Time')
plt.ylabel('Total Sales')
```

Out[32]: Text(0, 0.5, 'Total Sales')



We can see that sales spike up during the December month and zero sale on Christmas Day.

In [33]:
```
# Let's look at the December month only

z_december = z[(z.index < "2019-01-01") & (z.index > "2018-11-30")]
z_december.head()
```

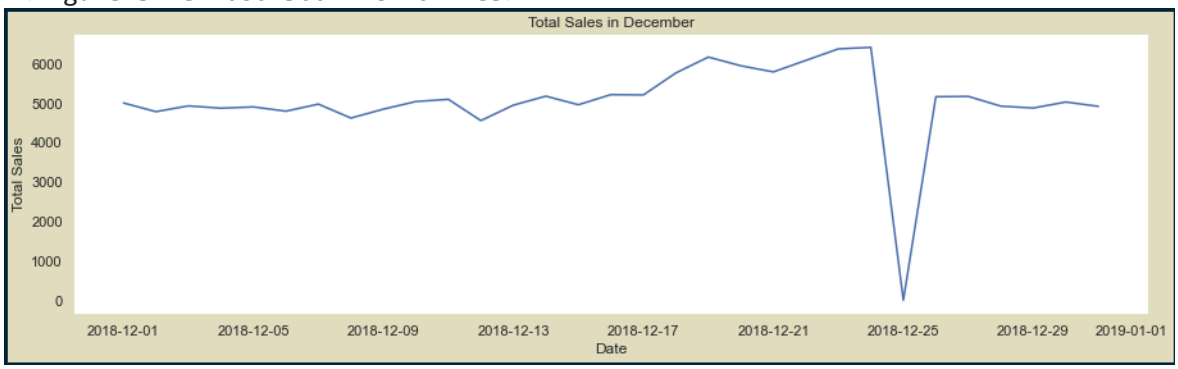
Out[33]:

Date	Total Sales
2018-12-01	5000.9
2018-12-02	4781.1
2018-12-03	4927.0
2018-12-04	4869.4
2018-12-05	4900.5

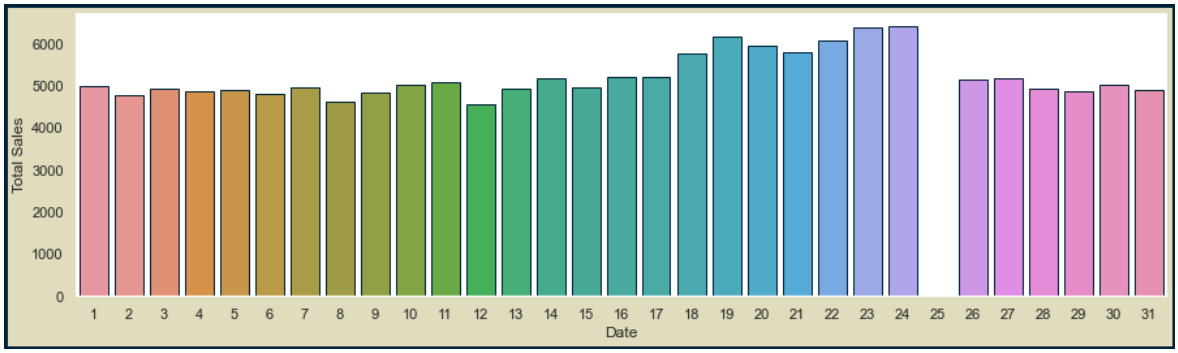
In [34]:
```
plt.figure(figsize = (15, 5))
fig = plt.figure(linewidth=5, edgecolor="#04253a", facecolor = '#e1ddbf')

ax = sns.lineplot(data= z_december, x= 'Date', y = 'Total Sales')
ax.set_facecolor("#ffffff")

plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('Total Sales in December')
```

Out[34]: Text(0.5, 1.0, 'Total Sales in December')

<Figure size 1080x360 with 0 Axes>



In [35]:
```
# Reset index

z_december.reset_index(drop = True, inplace = True)
z_december.head()

z_december['Date'] = z_december.index + 1
z_december.head()

fig = plt.figure(linewidth=5, edgecolor="#04253a", facecolor = '#e1ddbf')

ax = sns.barplot(x = 'Date', y ='Total Sales', data = z_december,  edgecolor=

ax.set_facecolor("#ffffff")
```



- We can see that the Sales have increased till the day before Christmas i.e. 2018-12-24 and there are no transaction records on 25th of December because of the Holiday and also the sales went down after Christmas.

Data analysis on customer segments

1. Who spends the most on chips i.e. describing customers by lifestage and premimum category?

In [36]:
```
df4
```

Out[36]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	P
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips 160g	
3	2019-03-09	1	1307	347	54	CCs Original 175g	
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
...	...	...	...	...	...	...	
246199	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
246200	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	
246201	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	
246202	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
246203	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	

246204 rows × 12 columns

In [37]:
```
df4['LIFESTAGE'].value_counts()
```

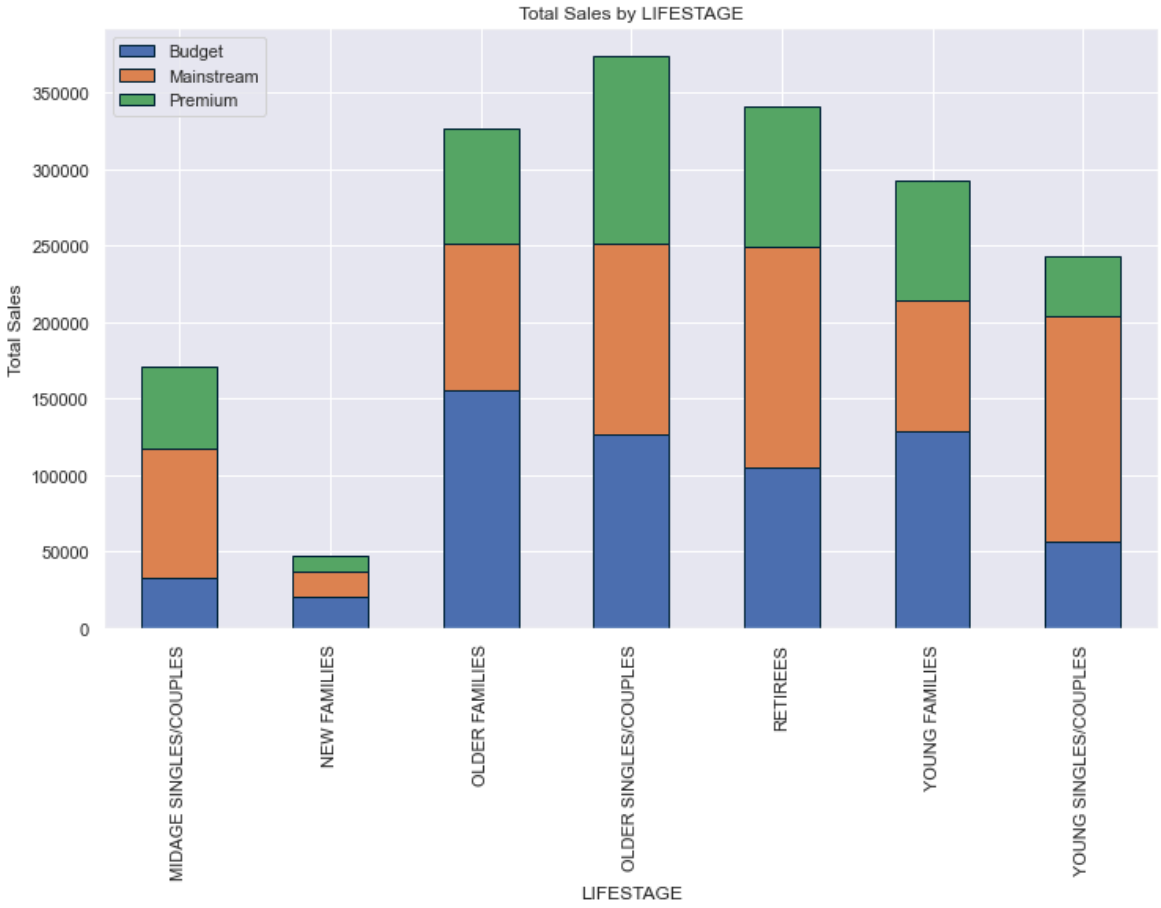
Out[37]:
```
OLDER SINGLES/COUPLES     50677
RETIREES                  46342
OLDER FAMILIES            45042
YOUNG FAMILIES            40395
YOUNG SINGLES/COUPLES     33917
MIDAGE SINGLES/COUPLES    23342
NEW FAMILIES               6489
Name: LIFESTAGE, dtype: int64
```

In [38]:
```
df4['PREMIUM_CUSTOMER'].value_counts()
```

Out[38]:
```
Mainstream    94839
Budget        86567
Premium       64798
Name: PREMIUM_CUSTOMER, dtype: int64
```

In [39]:
```
df8 = pd.DataFrame(df4.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).TOT_SALES.s

fig = plt.figure(linewidth=5, edgecolor="#04253a", facecolor = '#e1ddbf')

df8.unstack().plot(kind = 'bar', stacked = True, figsize = (12, 7), title = '

plt.ylabel('Total Sales')
plt.legend(['Budget', 'Mainstream', 'Premium'], loc = 2)
```

Out[39]:
```
<matplotlib.legend.Legend at 0x143059724c0>
<Figure size 1080x288 with 0 Axes>
```



- *The sales are high for Budget-Older Families, Mainstream-young singles/couples, Mainstream - retirees and premium - Older Single/Couples.*

2. How many customers are there in each segment?

In [40]:
```
df9 = pd.DataFrame(df4.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE']).LYLTY_CARD_
df9.rename(columns = {'LYLTY_CARD_NBR': 'Number of Customers'}, inplace = Tru
```

In [41]:
```
df9 = df9.sort_values(by = 'Number of Customers', ascending = False).head(10)
```

In [42]:
```
df9
```

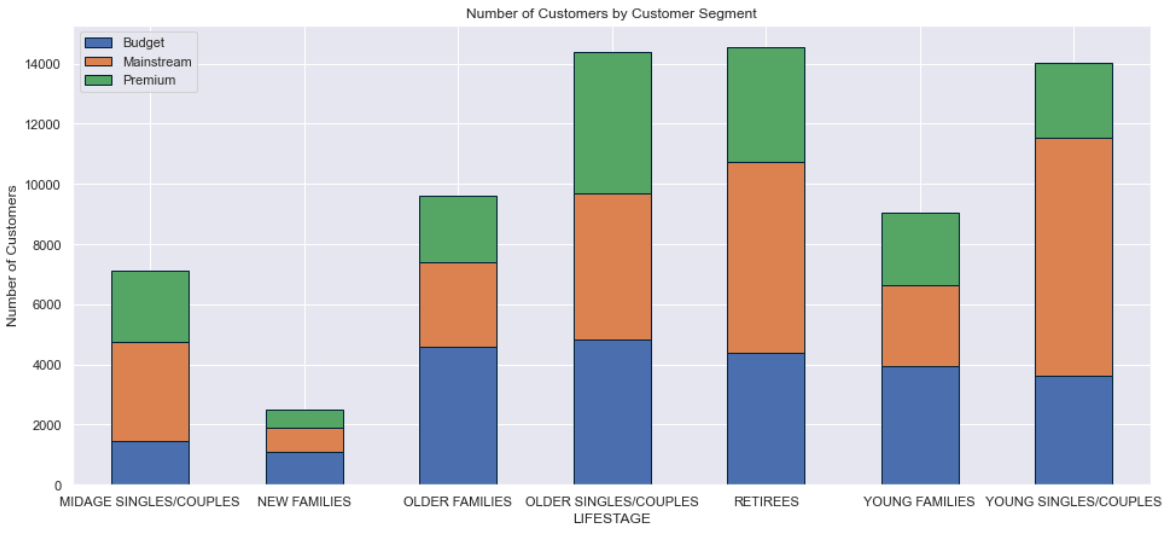
Out[42]:

		Number of Customers
PREMIUM_CUSTOMER	**LIFESTAGE**	
Mainstream	**YOUNG SINGLES/COUPLES**	7905
	RETIREES	6357
	OLDER SINGLES/COUPLES	4853
Budget	**OLDER SINGLES/COUPLES**	4846
Premium	**OLDER SINGLES/COUPLES**	4681
Budget	**OLDER FAMILIES**	4606
	RETIREES	4382
	YOUNG FAMILIES	3951
Premium	**RETIREES**	3811
Budget	**YOUNG SINGLES/COUPLES**	3644

In [43]:
```
df9 = pd.DataFrame(df4.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).LYLTY_CARD_
```

In [44]:
```
df9.unstack().plot(kind='bar', stacked = True , rot=0 , figsize = (16, 7), ti
plt.ylabel('Number of Customers')
plt.legend(['Budget', 'Mainstream', 'Premium'], loc = 2)
```

Out[44]: <matplotlib.legend.Legend at 0x14305986910>



There are more mainstream young singles/couples and retirees. This contributes to to more chips sales in these segments however this is not the major driver for the budget older families segment.

3. How many chips are bought per customer by segment?

In [45]:
```
# Average units per customer by PREMIUM_CUSTOMER and LIFESTAGE

df10 = df4.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE']).PROD_QTY.sum() / df4.gr

df10 = pd.DataFrame(df10, columns = {'Average Unit per Customer'})
```

In [46]:
```
df10.sort_values(by = 'Average Unit per Customer', ascending = False).head()
```

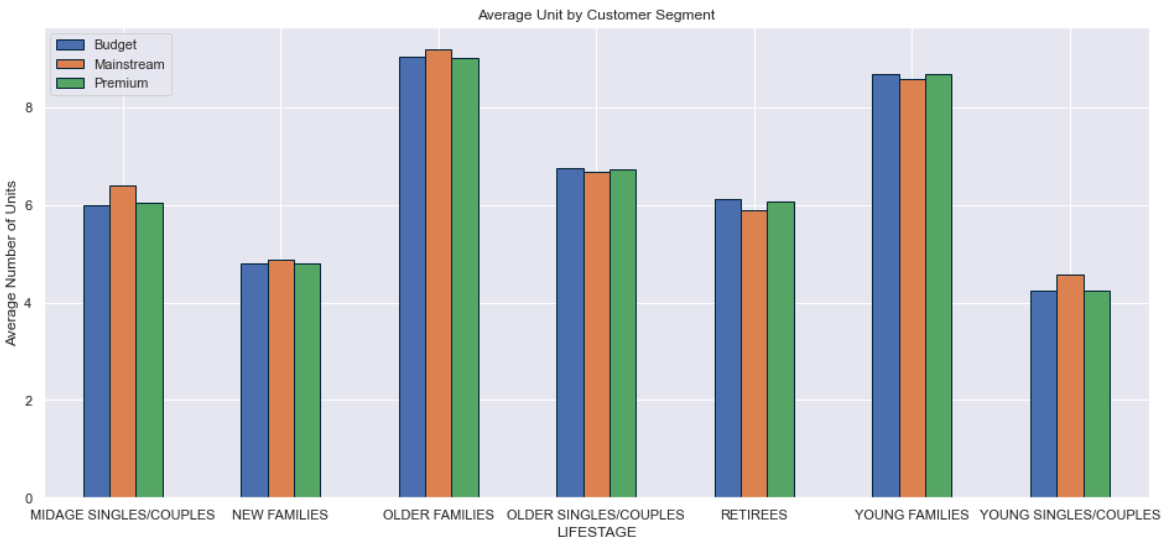
Out[46]:

		Average Unit per Customer
PREMIUM_CUSTOMER	**LIFESTAGE**	
Mainstream	**OLDER FAMILIES**	9.195839
Budget	**OLDER FAMILIES**	9.038645
Premium	**OLDER FAMILIES**	9.013453
Budget	**YOUNG FAMILIES**	8.677803
Premium	**YOUNG FAMILIES**	8.674593

In [47]:
```
df10 = pd.DataFrame(df4.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).PROD_QTY.s
```

In [48]:
```
df10.unstack().plot(kind = 'bar', figsize = (16, 7), rot = 0, title = 'Averag
plt.ylabel('Average Number of Units')
plt.legend(['Budget', 'Mainstream', 'Premium'], loc = 2)
```

Out[48]:
```
<matplotlib.legend.Legend at 0x14311bace80>
```



- For all the three Lifestages, Older families and Young Families buy more chips per customer.

4. What's the average chip price by customer segment?

In [49]:
```
# Average price per unit by PREMIUM_CUSTOMER and LIFESTAGE
```

```
df11 = df4.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE']).TOT_SALES.sum() / df4.g
df11 = pd.DataFrame(df11, columns = {'Price per Unit'})
```

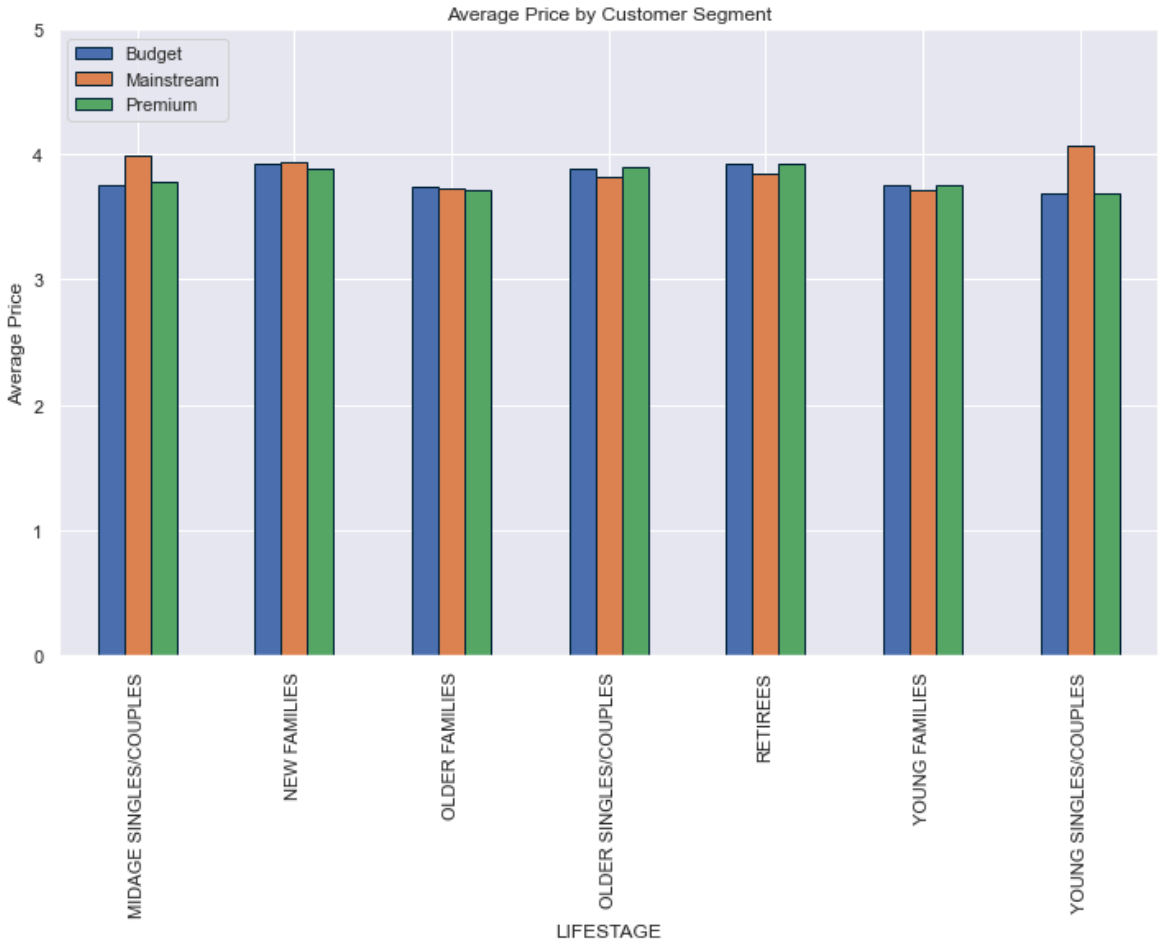
In [50]:
```
df11.sort_values(by = 'Price per Unit', ascending = False).head()
```

Out[50]:

		Price per Unit
PREMIUM_CUSTOMER	**LIFESTAGE**	
Mainstream	**YOUNG SINGLES/COUPLES**	4.072432
	MIDAGE SINGLES/COUPLES	3.991286
	NEW FAMILIES	3.936082
Budget	**NEW FAMILIES**	3.929084
	RETIREES	3.928989

In [51]:
```
# Visualise

df11 = pd.DataFrame(df4.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).TOT_SALES.
df11.unstack().plot(kind = 'bar', figsize = (12, 7), title = 'Average Price b
plt.ylabel('Average Price')
plt.legend(['Budget', 'Mainstream', 'Premium'], loc = 2)
```

Out[51]:
```
<matplotlib.legend.Legend at 0x14311e31880>
```



- Mainstream midage singles/couples and young singles/couples pay more per packet of chips compared to other segments.

In [52]:
```
# Perform an independent t-test between mainstream vs non-mainstream midage a

# Create a new dataframe pricePerUnit
pricePerUnit = df4

# Create a new column under pricePerUnit called PRICE
pricePerUnit['PRICE'] = pricePerUnit['TOT_SALES'] / pricePerUnit['PROD_QTY']

# Let's have a look
pricePerUnit.head()
```

Out[52]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_(
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips 160g	
3	2019-03-09	1	1307	347	54	CCs Original 175g	
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	

In [53]:
```
# Let's group our data into mainstream and non-mainstream

mainstream = pricePerUnit.loc[(pricePerUnit['PREMIUM_CUSTOMER'] == 'Mainstrea
nonMainstream = pricePerUnit.loc[(pricePerUnit['PREMIUM_CUSTOMER'] != 'Mainst
```

In [54]:
```
# Perform t-test

ttest_ind(mainstream, nonMainstream)
```

Out[54]:
```
Ttest_indResult(statistic=37.752225863415426, pvalue=4.307427069453844e-308)
```

Let us further explore and target the segment Mainstream and young singles/couples that contributes most to the sales.

Let's find out if they tend to buy a particular brand of chips.

In [55]:
```
df4
```

Out[55]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	P
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips 160g	
3	2019-03-09	1	1307	347	54	CCs Original 175g	
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
...	...	...	...	...	...	...	
246199	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
246200	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	
246201	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	
246202	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
246203	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	

246204 rows × 13 columns

In [56]:
```
dfa = df4[(df4['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') & (df4['PREMIUM_CUSTO
dfb = df4[(df4['LIFESTAGE'] != 'YOUNG SINGLES/COUPLES') & (df4['PREMIUM_CUSTO
```

In [57]:
```
dfa_quantity = dfa['PROD_QTY'].sum()
dfa_quantity
```

Out[57]: 36074

In [58]:
```
dfb_quantity = dfb['PROD_QTY'].sum()
dfb_quantity
```

Out[58]:
```
262133
```

In [59]:
```
dfa_quantity_brand = dfa.groupby(['Brand Name'])['PROD_QTY'].sum()
dfa_quantity_brand
```

Out[59]:
```
Brand Name
Burger        106
CCs           405
Cheetos       291
Cheezels      651
Cobs         1609
Dorito        569
Doritos      3867
French        143
Grain        1055
GrnWves       125
Infuzions    1797
Infzns        541
Kettle       7106
NCC           132
Natural       578
Pringles     4306
RRD          1582
Smith         239
Smiths       3240
Snbts         126
Sunbites      104
Thins        2182
Tostitos     1640
Twisties     1664
Tyrrells     1143
WW            873
Name: PROD_QTY, dtype: int64
```

In [60]:
```
dfb_quantity_brand = dfb.groupby(['Brand Name'])['PROD_QTY'].sum()
dfb_quantity_brand
```

Out[60]:
```
Brand Name
Burger        1723
CCs           4861
Cheetos       3094
Cheezels      4926
Cobs         10085
Dorito        3379
Doritos      23079
French        1504
Grain         6525
GrnWves       1618
Infuzions    11768
Infzns        3290
Kettle       43564
NCC           1587
Natural       6522
Pringles     26502
RRD          17671
Smith         3259
Smiths       29344
Snbts         1780
Sunbites      1544
Thins        14999
Tostitos     10032
Twisties      9903
Tyrrells      6727
WW           12847
Name: PROD_QTY, dtype: int64
```

In [61]:
```
# To check the brand affinity

dfa_affinity = dfa_quantity_brand/dfa_quantity
dfb_affinity = dfb_quantity_brand/dfb_quantity

affinity = (dfa_affinity/dfb_affinity).sort_values(ascending = False)

df_affinity= pd.DataFrame({'Brand Name':affinity.index, 'Affinity':affinity.v
df_affinity
```

Out[61]:

	Brand Name	Affinity
0	Tyrrells	1.234674
1	Dorito	1.223634
2	Twisties	1.220995
3	Doritos	1.217544
4	Infzns	1.194892
5	Tostitos	1.187911
6	Kettle	1.185291
7	Pringles	1.180654
8	Grain	1.174896
9	Cobs	1.159331
10	Infuzions	1.109616
11	Thins	1.057109
12	Cheezels	0.960316
13	Smiths	0.802330
14	French	0.690901
15	Cheetos	0.683440
16	RRD	0.650538
17	Natural	0.643983
18	CCs	0.605420
19	NCC	0.604400
20	GrnWves	0.561383
21	Smith	0.532894
22	Snbts	0.514373
23	WW	0.493787
24	Sunbites	0.489456
25	Burger	0.447042

- Tyrell chips seems to be the most popular brand for Mainstream young singles/couples almost 23% more than the rest of the audience whereas Burger Rings were purchased less than 56% less as compared to rest of the people.

Let's also find out if our target segment tends to buy larger packs of chips.

In [62]:
```
dfa_quantity_pack = dfa.groupby(['pack_size'])['PROD_QTY'].sum()
dfb_quantity_pack = dfb.groupby(['pack_size'])['PROD_QTY'].sum()
```

In [63]:
```
# To Check Pack Affinity
dfa_affinity_pack = dfa_quantity_pack/dfa_quantity
dfb_affinity_pack = dfb_quantity_pack/dfb_quantity

affinity_pack = (dfa_affinity_pack/dfb_affinity_pack).sort_values(ascending =

df_affinity_pack = pd.DataFrame({'Pack Size':affinity_pack.index, 'Affinity':
df_affinity_pack
```

Out[63]:

	Pack Size	Affinity
0	270	1.276508
1	380	1.252702
2	330	1.210177
3	110	1.187112
4	134	1.180654
5	210	1.174896
6	135	1.127825
7	250	1.113282
8	170	1.008014
9	150	0.962471
10	175	0.940317
11	165	0.905054
12	190	0.616927
13	180	0.561383
14	160	0.523389
15	125	0.502890
16	90	0.502799
17	200	0.485132
18	70	0.482680
19	220	0.447042

In [64]:
```
Product_name = df4[(df4['pack_size'] == '270')]['PROD_NAME']
```

```
Product_name.unique()
```

```
Out[64]: array(['Twisties Cheese     270g', 'Twisties Chicken270g'], dtype=object)
```

- Mainstream young singles/couples are 27% more likely to purchase a 270g pack of Twisties Cheese chips which reflects the higher volume of sales.

```
In [65]: Product_name_least = df4[(df4['pack_size'] == '220')]['PROD_NAME']

Product_name_least.unique()
```

```
Out[65]: array(['Burger Rings 220g'], dtype=object)
```

- Mainstream young singles/couples are 56% less likely to purchase a 220g pack of Burger Rings which reflects the lower volume of sales.

Conclusion

- Budget – older families, Mainstream – young singles/couples, and Mainstream – retirees shoppers have accounted for the majority of sales.
- Mainstream young singles/couples and retirees spend more money on chips than other purchasers mainly due to the fact that there are more customers in these segments.
- Additionally, young, middle-aged, and mainstream individuals and couples are more inclined to pay more for each bag of chips.
- They are also 23% more likely to purchase 'Tyrrells' and '270g' pack sizes than the rest of the population.